Weekly Report

08/16/2015-08/23/2015

Research

In our work, we plan to send the map image to background. There are several methods to capture screenshot of a website. But few methods meet our requirement.

BaiduMap API

BaiduMap provide a static map api, we can get a image in png format by sending a http request[Fig.1].

http://api.map.baidu.com/staticimage
?center=116.403874,39.914888&width=300&height=200&zoom=11

Figure 1: Baidu Static Map

However, we can't define the mapstyle(hide poi, road and building) in http request(not supported by BaiduMap api). Boundary computing benefit from a clear map showing Greenland or manmade only.

Html2canvas.js

Html2canvas.js allows us to take screenshots of webpages. Html2canvas will read DOM of a web and rebuild a picture including the DOM. Html2canvas does not make an actual screenshot, but builds the screenshot based on the information on the page. Only the standard html elements will be recognized and showed in "screenshot". So html2canvas is not suit to BaiduMap api. Due to security reasons, javascript is not allowed to capture a real screenshot in almost all browsers.

PhantomJS

PhantomJS is a browser based on webkit without interface. We can run it in command line only. Since PhantomJS is using WebKit, it can open a web page save as an image.

```
var page = require('webpage').create();
page.open('http://github.com/', function() {
  page.render('github.png');
  phantom.exit();
});
```

We are supposed to set mapstyle to BaiduMap in javascript(supported in BaiduMap api) first. Then we should find the Greenland we need and click a button that send information(location and mapstyle) to background service. Java program will download a screenshot in command line using PhantomJS and compute boundary according to screenshot. Capture screenshots takes a few seconds when PhantomJS needs open a new website and download map data. Sometimes we will get a screenshot with blank block because of failure in loading map data

Google Chrome Extensions

Our chrome extension consists of two javascript documents(background.js and contentscript.js) which are allowed to use chrome apis such as capture screenshots. Cententscript.js is javascript files that run in the context of web pages and can read DOM of the web page. background.js contentscript.js and javascript in our website are pairwise independence which means we can't use chrome apis in our javascript directly and can't use our javascript function in chrome extension directly. The only way to connect these js files is passing messages. A EventListener is add to a button by contenscript.js when a web page is opening. The button send a message to chrome's background after clicked. After background.js get the message, it will take a screenshot and send image data back to web page.

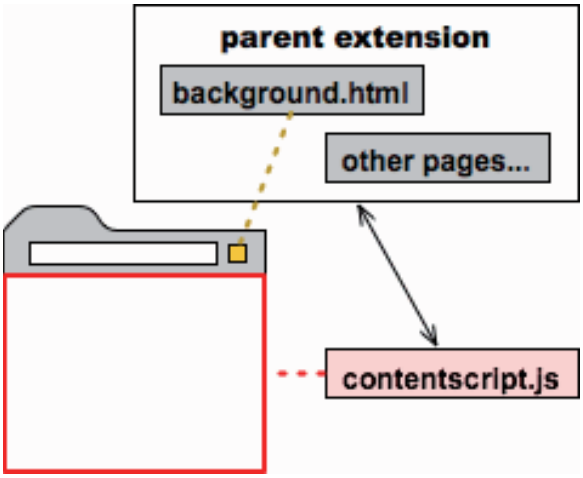

Figure 2: JS files

```
%contenscript.js
document.getElementById("theButton").addEventListener("mousedown",
  function() {
    console.log("click!");
    chrome.runtime.sendMessage(
    {greeting: "hello"}, function(response) {
      console.log(response.farewell);
    });
 }
, "*");

%background.js
chrome.runtime.onMessage.addListener(
  function(request, sender, sendResponse) {
    if (request.greeting == "hello")
    {
      chrome.tabs.captureVisibleTab(function(screenshotUrl) {
        console.log(screenshotUrl);
        chrome.tabs.executeScript({
          code:
          "document.getElementById('target').src='" + screenshotUrl + "'"
        });
      });
```

```
        sendResponse({farewell: "goodbye"});
      }
    }
);
```

Plan for next week

- Get boundary of greenland of five cities using our website.